Homework 4

Christophe Hunt

February 20, 2017

Contents

1 Page 191: problem 3

Using Monte Carlo Simulation, write an algorithm to calculate an approximation to π by considering the number of random points selected inside the quarter circle

$$Q : x^2 + y^2 = 1, x \geq 0, y \geq 0$$

where the quarter circle is taken to be inside the square

$$S : 0 \leq x \leq 1 \; and \; 0 \leq y \leq 1$$

Use the equation $\frac{\pi}{4} = area; \frac{Q}{area} S$.

```
set.seed(1234)

monte_carlo <- function(n){
  counter = 0
  for (i in 1:n){
    y <- runif(1, 0, 1)
    x <- runif(1, 0, 1)
    if ((x^2 + y^2) < 1){
      counter <- counter + 1
    } else {
      counter <- counter
    }
  }
  return(counter)
}
```

```
n <- 500
(monte_carlo(n)/n)*4
```

```
## [1] 3.104
```

```
n <- 50000
(monte_carlo(n)/n)*4
```

```
## [1] 3.14312
```

The results of the algorithm get us very close to the value of π which is 3.1415927 as calculated in R.

2 Page 194: problem 1

Use the middle-square method to generate.

```
middle_square <- function(n, seed) {
          suppressMessages(require(stringr))
          results <- list("instances" = n, "starting_seed" = seed)
          for (i in 1:n){
            j <- (seed^2)
              if (nchar(j) < 8){
                  j <- str_pad(j, 8, pad = "0")
              } else if (nchar(j) > 8) {
                j <- substr(j, 1, 8)
              }
            t <- j
            j <- substr(j, 3, 6)
            seed <- as.numeric(j)
            results[[length(results)+1]] <- list("number" = t, "new_seed" = seed)
          }
         return(results)
}
```

2.1 a. 10 random numbers using $x_0 = 1009$

```
middle_square(10, 1009)
```

$instances [1] 10

$starting_seed [1] 1009

[[3]][[3]]$number [1] “01018081”

[[3]]$new_seed [1] 180

[[4]][[4]]$number [1] “00032400”

[[4]]$new_seed [1] 324

[[5]][[5]]$number [1] “00104976”

[[5]]$new_seed [1] 1049

[[6]][[6]]$number [1] “01100401”

[[6]]$new_seed [1] 1004

[[7]][[7]]$number [1] “01008016”

[[7]]$new_seed [1] 80

[[8]][[8]]$number [1] “00006400”

[[8]]$new_seed [1] 64

[[9]][[9]]$number [1] “00004096”

[[9]]$new_seed [1] 40

[[10]][[10]]$number [1] “00001600”

[[10]]$new_seed [1] 16

[[11]][[11]]$number [1] “00000256”

[[11]]$new_seed [1] 2

[[12]][[12]]$number [1] “00000004”

[[12]]$new_seed [1] 0

2.2 b. 20 random numbers using $x_0 = 653217$

```
middle_square(20, 653217)
```

$instances [1] 20

$starting_seed [1] 653217

[[3]][[3]]$number [1] “42669244”

[[3]]$new_seed [1] 6692

[[4]][[4]]$number [1] 44782864

[[4]]$new_seed [1] 7828

[[5]][[5]]$number [1] 61277584

[[5]]$new_seed [1] 2775

[[6]][[6]]$number [1] “07700625”

[[6]]$new_seed [1] 7006

[[7]][[7]]$number [1] 49084036

[[7]]$new_seed [1] 840

[[8]][[8]]$number [1] “00705600”

[[8]]$new_seed [1] 7056

[[9]][[9]]$number [1] 49787136

[[9]]$new_seed [1] 7871

[[10]][[10]]$number [1] 61952641

[[10]]$new_seed [1] 9526

[[11]][[11]]$number [1] 90744676

[[11]]$new_seed [1] 7446

[[12]][[12]]$number [1] 55442916

[[12]]$new_seed [1] 4429

[[13]][[13]]$number [1] 19616041

[[13]]$new_seed [1] 6160

[[14]][[14]]$number [1] 37945600

[[14]]$new_seed [1] 9456

[[15]][[15]]$number [1] 89415936

[[15]]$new_seed [1] 4159

[[16]][[16]]$number [1] 17297281

[[16]]$new_seed [1] 2972

[[17]][[17]]$number [1] “08832784”

[[17]]$new_seed [1] 8327

[[18]][[18]]$number [1] 69338929

[[18]]$new_seed [1] 3389

[[19]][[19]]$number [1] 11485321

[[19]]$new_seed [1] 4853

[[20]][[20]]$number [1] 23551609

[[20]]$new_seed [1] 5516

[[21]][[21]]$number [1] 30426256

[[21]]$new_seed [1] 4262

[[22]][[22]]$number [1] 18164644

[[22]]$new_seed [1] 1646

2.3 c. 15 random numbers using $x_0 = 3043$

```
middle_square(15, 3043)
```

$instances [1] 15

$starting_seed [1] 3043

[[3]][[3]]$number [1] “09259849”

[[3]]$new_seed [1] 2598

[[4]][[4]]$number [1] “06749604”

[[4]]$new_seed [1] 7496

[[5]][[5]]$number [1] 56190016

[[5]]$new_seed [1] 1900

[[6]][[6]]$number [1] “03610000”

[[6]]$new_seed [1] 6100

[[7]][[7]]$number [1] 37210000

[[7]]$new_seed [1] 2100

[[8]][[8]]$number [1] “04410000”

[[8]]$new_seed [1] 4100

[[9]][[9]]$number [1] 16810000

[[9]]$new_seed [1] 8100

[[10]][[10]]$number [1] 65610000

[[10]]$new_seed [1] 6100

[[11]][[11]]$number [1] 37210000

[[11]]$new_seed [1] 2100

[[12]][[12]]$number [1] “04410000”

[[12]]$new_seed [1] 4100

[[13]][[13]]$number [1] 16810000

[[13]]$new_seed [1] 8100

[[14]][[14]]$number [1] 65610000

[[14]]$new_seed [1] 6100

[[15]][[15]]$number [1] 37210000

[[15]]$new_seed [1] 2100

[[16]][[16]]$number [1] “04410000”

[[16]]$new_seed [1] 4100

[[17]][[17]]$number [1] 16810000

[[17]]$new_seed [1] 8100

2.4 d. Comment about the results of each sequence. Was there cycling? Did each sequence degenerate rapidly?

The first sequence degenerated very rapidly. It reached 0 by the 10 iteration and would not be suitable for use in a very sensitive analysis. It is concerning that the algorithm can reach 0 so quickly. Also, in c. we see the values begin repeating and this again is very concerning for a sensitive analysis. It may be difficult to spot this type of cycling in a simulation and may provide inaccurate results. However, in b. we do see the power of the randomness generation in the algorithm. As the literature says the random generation by this method is no longer a reliable algorithm.

3 Page 199: problem 4

Given loaded dice according to the following distribution, use Monte Carlo simulation to simulate the sum of 300 rolls of two unfair dice.

```
library(knitr)
Roll <- c(1:6)
Die_1 <- c(.1,.1,.2,.3,.2,.1)
Die_2 <- c(.3,.1,.2,.1,.05,.25)
kable(as.data.frame(cbind(Roll, Die_1, Die_2)))
```

Roll	Die_1	Die_2
1	0.1	0.30
2	0.1	0.10
3	0.2	0.20
4	0.3	0.10
5	0.2	0.05
6	0.1	0.25

```
unfair_roll <- function(n, probs){
    if (length(probs) != 6){
      print("missing probabilities for all 6 sides")
      break
    }
    results <- list("1" = 0, "2" = 0 , "3" = 0, "4" = 0, "5" = 0, "6" = 0)
         for (i in 1:n){
       j <- 1
       x <- runif(j, 0, j)
       if (x == 0){
        results[[j]] <- results[[j]] + 1
       } else if (x > 0 & x <= probs[[j]]){
        results[[j]] <- results[[j]] + 1
        next
       } else {
       j <- j + 1
       while ((x > sum(probs[0:(j-1)]) & x <= sum(probs[0:j])) != TRUE) {
       j <- j + 1
       }
       results[[j]] <- results[[j]] + 1
       }
     }
    for (i in 1:6){
     names(results[[i]]) <- "occurences"
     results[[i]][["probability"]] <- round(results[[i]]/n,2)
    }
  return(results)
}
```

```
unfair_roll(300, Die_1)
```

```
## $`1`
##  occurences probability
##       27.00        0.09
```

```
##
## $`2`
##  occurences probability
##       34.00        0.11
##
## $`3`
##  occurences probability
##       56.00        0.19
##
## $`4`
##  occurences probability
##        90.0         0.3
##
## $`5`
##  occurences probability
##        59.0         0.2
##
## $`6`
##  occurences probability
##       34.00        0.11
```

```
unfair_roll(300, Die_2)
```

```
## $`1`
##  occurences probability
##       82.00        0.27
##
## $`2`
##  occurences probability
##       24.00        0.08
##
## $`3`
##  occurences probability
##       69.00        0.23
##
## $`4`
##  occurences probability
##       27.00        0.09
##
## $`5`
##  occurences probability
##       12.00        0.04
##
## $`6`
##  occurences probability
##       86.00        0.29
```

The simulation provides rolls of the dice as we would expect. The probability is very similar to the distribution provided and is a very useful technique to generate a simulation of random rolls of a dice.

4 Page 211: problem 3

In many situations, the time T between deliveries and the order quantity Q is not fixed. Instead, an order is placed for a specific amount of gasoline. Depending on how many orders are placed in a given time interval, the time to fill an order varies. You have no reason to believe that the performance of the delivery operation will change. Therefore, you have examined records for the past 100 deliveries and found the following lag times, or extra days, required to fill your order:

```
Lag_time <- c(2:7)
number_of_occurrences <- c(10, 25, 30, 20, 13, 2)
X <- as.data.frame(cbind(Lag_time, number_of_occurrences))
kable(rbind(X, c(" ", sum(number_of_occurrences))))
```

Lag_time	number_of_occurrences
2	10
3	25
4	30
5	20
6	13
7	2
	100

Construct a Monte Carlo simulation for the lag time sub model. If you have a handheld calculator or computer available, test your sub model by running 1000 trials and comparing the number of occurrences of the various lag times with the historical data.

```
Lag_time <- c(2:7)
number_of_occurrences <- c(10, 25, 30, 20, 13, 2)

lag_time <- function(n, probs, outcomes){
        results <- list()
        for (i in 1:length(outcomes)){
         results[[i]] <- 0
         names(results[[i]]) <- (paste("# of deliveries on day", outcomes[[i]]))
        }
    for (i in 1:n){
      j <- 1
      x <- runif(j, 0, j)
      if (x == 0){
       results[[j]] <- results[[j]] + 1
      } else if (x > 0 & x <= probs[[j]]){
       results[[j]] <- results[[j]] + 1
       next
      } else {
      j <- j + 1
      while ((x > sum(probs[0:(j-1)]) & x <= sum(probs[0:j])) != TRUE) {
      j <- j + 1
      }
      results[[j]] <- results[[j]] + 1
      }
    }
   for (i in 1:length(results)){
    results[[i]][["probability"]] <- round(results[[i]]/n,2)
```

```
    }
  return(results)
}
probs <- (number_of_occurrences/sum(number_of_occurrences))
lag_time(1000, probs = probs , outcomes = Lag_time)
```

```
## [[1]]
## # of deliveries on day 2             probability
##                    102.0                     0.1
##
## [[2]]
## # of deliveries on day 3             probability
##                   237.00                    0.24
##
## [[3]]
## # of deliveries on day 4             probability
##                    299.0                     0.3
##
## [[4]]
## # of deliveries on day 5             probability
##                   211.00                    0.21
##
## [[5]]
## # of deliveries on day 6             probability
##                   125.00                    0.12
##
## [[6]]
## # of deliveries on day 7             probability
##                    26.00                    0.03
```

The results follow the distribution provided, it is easy to compare in this situation because the total observations is 100 making the probabilities quick to spot. It is a useful tool to visualize how many deliveries over the course of 1000 may fall into the lag range and we can plan accordingly.